Eric Wolf, 3.11.2019

Rominos

Lösungsideen

Berechnen der Rominos

Für die Berechnung der Rominos entwickelte ich ein rekursives Verfahren, bei dem von dem Rominos eines bestimmten Grades so lange die Rominos des nächsten Grades errechnet werden, bis man den gewünschten Grad erreicht.

Startwert

Als Startwert benutze ich die Rominos des zweiten Grades, weil dies der kleinste Grad ist, für welchen Rominos existieren.

Dies liegt an der Eigenschaft der Rominos, dass sie immer ein „Romino-Paar“ besitzen, welches aus mindestens zwei Rominos bestehen muss.

Diese Anzahl an Rominos ist unterhalb des Zweiten Grades nicht mehr gegeben, weswegen dort keine Rominos existieren können.

Berechnen des nächsten Grades

Das Berechnen des nächsten Grades wird durch das Hinzufügen eines Quadrates an einer beliebigen freien Stelle, welche ein Quadrat als direkten oder diagonalen Nachbar besitzt, realisiert.

Wenn die resultierende Figur ein Romino Paar besitzt, das existierende also nicht durch das neue Quadrat zerstört wurde oder ein neues gebildet wurde, wird sie also neues Romino gespeichert.

Dieser Vorgang wird für jede freie Stelle, welche den obigen Kriterien entspricht, wiederholt.

Entfernen von Duplikaten

Zur Entfernung von Duplikaten werden zuerst alle möglichen Duplikate berechnet, welche dann mit den Rominos verglichen werden.

Wenn sich ein Duplikat bereits unter den Rominos befindet wird das Romino, für welches die Duplikate berechnet wurden, verworfen.

Eric Wolf, 3.11.2019

Berechnung der Gespiegelten und gedrehten Duplikate

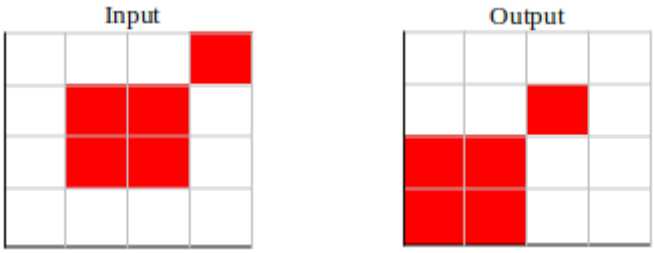
Für die gespiegelten und gedrehten Duplikate werden zuerst für das Romino die 3 gedrehten Varianten (+ 90°, + 180°, - 90°) berechnet.

Danach wird für jede dieser Varianten, inklusive des ursprünglichen Rominos, die 3 Spiegelungen (Y Achse, X Achse, Spiegelung 1 an X Achse) berechnet, weswegen am Ende 16 mögliche Duplikate (4 Drehungen mal 4 Spiegelungen) entstehen.

Man kann auch zuerst die Spiegelungen und danach die Drehungen berechnen, was das selbe Ergebnis bringen wird.

Berechnung der Verschobenen Duplikate

Verschobene Duplikate werden durch ein Anschmiegen an die X und Y Achse ausgeschlossen, weil Duplikate in verschiedenen Postionen mit gleicher Form so immer den selben Platz einnehmen werden.



Implementation

Berechnen der Rominos

Startwert

Der Startwert besteht wie oben genannt aus dem einzigen Romino zweiten Grades.

Berechnen des nächsten Grades

Die einzelnen Rominos werden durch Instanzen der Klasse „Romino“ repräsentiert, welche die Methode „upgrade“ enthält.

Diese Methode generiert aus dem Romino des Grades n mehrere Rominos des Grades n+1, indem an jede freie Stelle, welche direkt an das Romino angrenzt, ein neues Quadrat platziert wird, wobei pro neuem Romino nur ein Quadrat hinzugefügt wird.

Zudem wird überprüft, ob die neue Form ein Romino ist, sie also ein Romino Paar enthält, um Figuren, wo das Paar invalidiert wurde, herauszufiltern.

Die Positionen, welche besetzt werden könnten ohne eine Figur ohne Romino Paar zu erzeugen, hätten auch vorher berechnet werden können, was mir jedoch zu kompliziert war und wahrscheinlich die Laufzeit nur minimal verkürzt hätte.

Eric Wolf, 3.11.2019

Entfernen von Duplikaten

Duplikate werden wie oben genannt berechnet, für die Überprüfung, ob dieses Romino bereits existiert, wird jedoch der Datentyp „set“ verwendet, weil dieser durch eine Hashtabelle Duplikate verhindert.

Um die Instanzen der Klasse Romino hashbar zu machen besitzt sie die Methoden __eq__ und __hash__, welche das Vergleichen und Hashen der Instanzen erlaubt.

Die Methode __eq__ überprüft ob das zu vergleichende Objekt eine Instanz der Klasse Romino ist und ob die Varianten, welche als „frozenset“, ein spezielle hashbare Variante des „set“, generiert werden, von sich und dem anderen Objekt gleich sind, was von „set“ zur Vermeidung von Hashkollisionen verwendet wird.

Die Varianten werden als frozenset von der Methode „get_variants“ generiert, weil set und frozenset keine Ordnung besitzen und so die Reihenfolge der Varianten egal ist.

Zudem werden sie nicht zwischengespeichert, weil Tests mit diesem Verhalten einen sehr hohen Arbeitsspeicherverbrauch zeigten (mehr als 10 GB bei n = 10).

Die Methode __hash__ erzeugt wiederum aus den eigenen Varianten einen Hash, welcher zur Speicherung in einem set benötigt wird.

Graphische Ausgabe

Für die graphische Ausgabe benutze ich das in Python enthaltene tkinter Modul, mit welchem per „–draw“ Kommandozeilenargument die Rominos in einem Fenster mit einer Größe von 1024x768 Pixeln (anpassbar mit „--resolution“) angezeigt werden können.

Dafür wird zuerst die Abmessungen des Fensters berechnet, welche von der Breite und Höhe eines Rominos, der Anzahl pro Zeile und der Anzahl der Zeilen abhängt.

Da ein Romino des Grades n maximal n Quadrate hoch oder breit ein kann und jedes Quadrat 10x10 Pixel groß ist bekommt jedes Romino ein Platz, welcher ein Quadrat mit der Kantenlänge n*10 darstellt.

Zusätzlich besitzen diese Quadrate oben und rechts einen 10 Pixel breiten leeren Rand, welcher eine klare Unterscheidung der einzelnen Rominos ermöglichen soll.

Die Anzahl der Rominos pro Reihe wird durch das Abrunden des Ergbnisses von Fensterbreite/Rominobreite ermittelt, weil ohne Abrunden ein Romino nur zum Teil auf die Reihe passen würde.

Zum Schluss wird die Anzahl der Reihen berechnet, indem die Anzahl der Rominos durch die Anzahl pro Reihe geteilt wird, wobei im Falle einer nicht komplett gefüllten letzten Reihe aufgerundet wird.

Danach wird mit „get_window“ ein Fenster erstellt, welches die berechneten Maße und einen Scrollbalken besitzt.

Daraufhin wird für jedes Romino die „draw“ Methode aufgerufen, wobei jedoch die die Position auf der Reihe (pre_x) für jedes Romino erhöht wird, um ein übermalen zu verhindern.

Wenn die Anzahl der gezeichneten Rominos die berechnete Anzahl von Rominos pro Reihe erreicht wird sie auf Null gesetzt und die Reihe (pre_y) erhöht.

Aufgrund der Positionen der Achsen bei tkinter sind alle Rominos gespiegelt, was jedoch durch den Umstand, dass Spiegelungen eines Rominos als Duplikate gelten, ignoriert werden kann.

Verbesserungsmöglichkeiten

Das momentane Programm benötigt zur Berechnung der Rominos des Grades 10 ungefähr 16 Minuten, was wahrscheinlich durch die fehlende Zwischenspeicherung und der daraus Resultierenden ständigen Neuberechnung der Varianten eines Rominos hervorgerufen wird.

Da das Zwischenspeichern jedoch sehr viel Arbeitsspeicher benötigt müsste man ein komplett anderes Verfahren verwenden.

Man könnte außerdem für die Berechnung der Rominos auf Bäume zurückgreifen, für welche mir jedoch Erfahrungen fehlen.

Beispiele

Hilfeseite für Kommandozeilenargumente

n = 3

n = 8, show steps

n = 5, gui

n < 2